(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
14 juin 2001 (14.06.2001)

PCT

(10) Numéro de publication internationale
WO 01/42887 A2

(51) Classification internationale des brevets[7]: G06F 1/00

(21) Numéro de la demande internationale:
PCT/FR00/03463

(22) Date de dépôt international:
8 décembre 2000 (08.12.2000)

(25) Langue de dépôt: français

(26) Langue de publication: français

(30) Données relatives à la priorité:
99/15791 10 décembre 1999 (10.12.1999) FR

(71) Déposant *(pour tous les États désignés sauf US)*: GEMPLUS [FR/FR]; Avenue du Pic de Bertagne, Parc d'Activités de Gèmenos, F-13420 Gèmenos (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)*: GRIMAUD, Gilles [FR/FR]; 7, rue Sainte Anne, F-59800 Lille (FR). HAGIMONT, Daniel [FR/FR]; 73 rue du 19 Mars 1962, F-38920 Crolles (FR). VANDEWALLE, Jean-Jacques [FR/FR]; 18, rue Bernardy, F-13001 Marseille (FR).

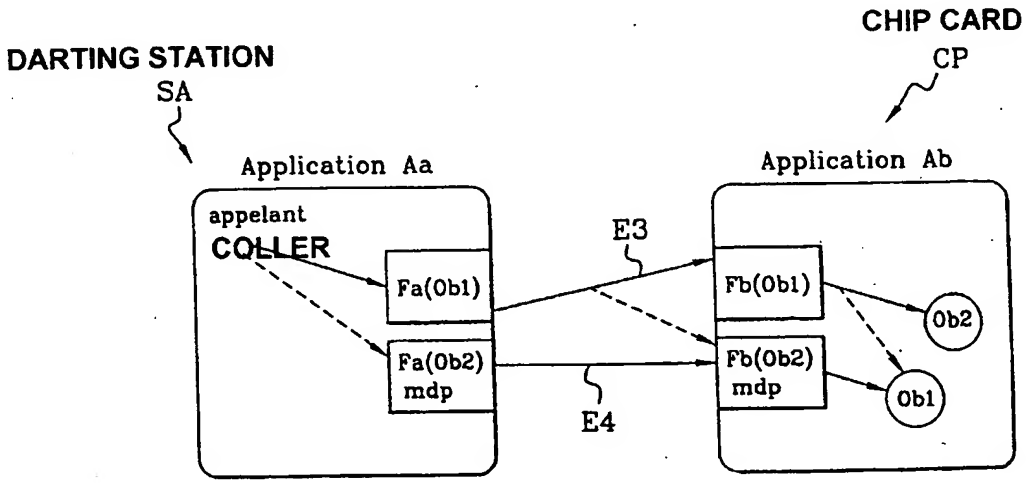
(81) États désignés *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) États désignés *(régional)*: brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU,

[Suite sur la page suivante]

(54) Title: CAPABILITY-BASED ACCESS CONTROL FOR APPLICATIONS IN PARTICULAR CO-OPERATING APPLICATIONS IN A CHIP CARD

(54) Titre: CONTRÔLE D'ACCES PAR CAPACITES POUR DES APPLICATIONS NOTAMMENT COOPERANTES DANS UNE CARTE A PUCE



(57) Abstract: The invention concerns an application programmer no longer responsible for managing access rights, the application code being independent of the protection in the chip card. The capability-based access control consists, when an application (Aa), for example in a docking station, is given access to an object (Ob1) pertaining to the other application (Ab) in a chip card (CP), in creating two capabilities (Fa(Ob1), Fb(Ob1)) respectively in the applications, as objects, to protect all subsequent accesses to the object by filtering them through the two capabilities. On accessing (E1) an object (Ob1) pertaining to an application (Ab), if a second object (Ob2) pertaining to the other application (Ab) is passed on to the latter, two other capabilities (Fa(Ob2, Fb(Ob2)) are added (E2) in the applications to protect access to the second object.

[Suite sur la page suivante]

WO 01/42887 A2

MC, NL, PT, SE, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Publiée:

— *Sans rapport de recherche internationale, sera republiée dès réception de ce rapport.*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé: Un programmeur d'applications ne se soucie plus de la gestion des droits d'accès, le code des applications étant indépendant de la protection dans une carte à puce. Le contôle d'accès par capacités comprend, lorsqu'à une application (Aa), par exemple dans une station d'accueil de carte, est donné un accès à un objet (Ob1) appartenant à l'autre application (Ab) dans une carte à puce (CP), la création de deux capacités (Fa(Ob1), Fb(Ob1)) respectivement dans les applications, en tant qu'objets, pour protéger tous les accès ultérieurs à l'objet en les filtrant au travers des deux capacités. Lors de l'accès (E1) à un objet (Ob1) appartenant à une application (Ab), si un deuxième objet (Ob2) appartenant à l'autre application (Ab) est passé à celle-ci, deux autres capacités (Fa(Ob2), Fb(Ob2)) sont ajoutées (E2) dans les applications pour protéger l'accès au deuxième objet.

Contrôle d'accès par capacités pour des applications notamment coopérantes dans une carte à puce

L'invention concerne les cartes à puce, dites également cartes à microcontrôleur ou cartes à circuit intégré, et plus généralement des moyens de traitement de données ouverts programmables pouvant être chargés d'applications écrites dans des langage de programmation évolués.

Une carte à puce ouverte, telle que présentée par exemple dans le document WO 98/19237, gère plusieurs applications, par exemple un compte client pour un magasin, un compte bancaire ou un porte-monnaie électronique. Certaines applications chargées dans la carte coopèrent parfois par exemple pour payer un achat du magasin, et/ou coopèrent également avec des applications s'exécutant hors de la carte.

La coopération des applications impose l'établissement de règles de droit d'accès, les applications ne se faisant pas forcément confiance. Par exemple, le compte client géré par le magasin ne doit pas s'approprier de données gérées par le porte-monnaie électronique.

Dans un environnement informatique, la gestion du contrôle d'accès consiste à associer des droits d'accès à des objets gérés dans l'environnement pour chaque usager, et à vérifier que ces droits d'accès sont respectés.

La gestion du contrôle d'accès est schématisée à la figure 1 par la gestion d'une matrice d'accès MA. Les lignes de la matrice MA correspondent aux droits de J objets O1 à OJ et les colonnes de cette matrice correspondent aux droits de I usagers U1 à UI. Une case de la matrice à l'intersection d'une ligne et

d'une colonne donne des droits d'accès Dij d'un usager Ui sur un objet Oj, avec 1 = i = I et 1 = j = J, par exemple un droit de lire, d'écrire ou d'exécuter un fichier.

En pratique, la matrice MA est partiellement vide, des usagers n'ayant souvent aucun droit sur de nombreux objets, et présente l'une des deux configurations consistant en un regroupement des droits d'accès par ligne et en un regroupement des droits d'accès par colonne.

Le regroupement par ligne revient à associer à chaque objet Oj, une liste d'accès indiquant les droits d'accès D1j à DIj sur l'objet respectivement pour les usagers U1 à UI. Dans une gestion par listes d'accès respectivement associées à des objets, seul le propriétaire d'un objet Oj modifie la liste d'accès D1j à DIj associée à l'objet ; une telle modification se fait explicitement en appelant une opération sur l'objet demandant la modification de sa liste d'accès. Les schémas de protection à base de listes d'accès sont alors qualifiés de statiques, la modification des droits d'accès étant complexe et les usagers ayant tendance à surdimensionner les droits d'accès de leurs objets. Ceci va à l'encontre du principe du moindre privilège (en anglais : *need to know principle*) selon lequel des droits d'accès ne sont accordés qu'au fur et à mesure des besoins.

Le regroupement par colonne associe à chaque usager Ui une liste de capacités indiquant les droits d'accès Di1 à DiJ de l'usager respectivement pour les objets sur lesquels l'usager possède un droit. Chaque élément (Oj, Dij) de la liste est appelé une capacité. Une capacité est un descripteur contenant l'identification d'un objet Oj ainsi qu'une définition de droits d'accès Dij sur cet objet. Dans

une gestion par capacités, un usager possède une liste de capacités, une capacité pouvant être comparée à un jeton donnant le droit de faire une opération sur l'objet. Une capacité identifie un objet, mais inclut en plus une définition des droits d'accès sur l'objet. Une capacité peut alors être utilisée comme un identificateur d'objet qu'une application peut passer en paramètre à une autre application, en suivant un mode de programmation classique, avec la limitation que cet identificateur n'autorise pas toutes les opérations sur l'objet désigné.

Il en découle que la modification des droits d'accès est plus naturelle avec des capacités : accorder à une application ou un usager des droits d'accès à un objet revient à lui passer en paramètre d'une opération l'identité de l'objet sous la forme d'une capacité.

Les capacités apportent une plus grande dynamique dans la gestion du contrôle d'accès, les droits d'accès pouvant être aisément échangés entre les usagers de l'environnement. Cependant, lorsqu'un usager ou une application doit passer en paramètre une capacité sur un objet, l'usager ou l'application doit auparavant décider des droits à transférer avec la capacité. Une opération permet en général de réduire les droits associés à la capacité si nécessaire, avant de la passer en paramètre.

Dans la technique antérieure, sont connues des réalisations matérielles et des réalisations logicielles des capacités comparables à des jetons. Les premières réalisations matérielles reposaient sur des machines spécialisées dans les années 70. Le mécanisme d'adressage de ces machines implantait directement la notion de capacité : un registre

d'adresse servant à adresser un objet contenait également les droits d'accès à l'objet (le registre contenant le jeton). Les valeurs de ces registres pouvaient être échangées entre les usagers, mais ne pouvaient pas être forgées, le matériel ne le permettant pas. Les réalisations logicielles, plus récentes dans les années 80, reposaient sur le chiffrement pour la protection des capacités. Une capacité était signée et ne pouvait être créée que par le propriétaire de l'objet.

Ainsi, le document US 5781633 illustre une technique antérieure permettant un filtrage, au moyen de capacités, d'échanges de références d'objets entre différents processus. Des procédés cryptographiques sont utilisés pour garantir principalement l'intégrité des références échangées. La coopération entre processus est réalisée par la transmission d'une vue réduite d'un objet d'un processus, à un second processus. Le filtre ainsi créé est implanté au sein du processus demandeur d'accès. Dans un contexte de type « processus mutuellement méfiants », le procédé divulgué par le document est inopérant. En effet le processus demandeur d'accès peut modifier à sa guise le filtre qui lui a été transmis et accéder ainsi à des méthodes qui lui sont pourtant interdites.

L'invention a trait plus particulièrement à un mécanisme de contrôle d'accès basé sur un schéma de protection par capacités pour gérer la coopération entre des applications dans le contexte d'une carte à puce. En effet, le contexte de la carte à puce se caractérise par des coopérations entre des applications non prévues à l'avance. Il est donc difficile de satisfaire un schéma de protection comme

les listes d'accès où les droits d'accès sont le plus souvent pré-établis. La dynamique du schéma à base de capacités est un réel besoin.

Dans les solutions actuelles dans le contexte de la carte à puce, un programmeur doit gérer des capacités "à la main" dans le code de l'application, ce qui conduit à une complexité de programmation des applications protégées.

L'invention implante un modèle à capacité dans le contexte de la carte à puce pour poursuivre deux objectifs :

- Préserver la simplicité de programmation du langage JAVA dans le contexte de la JAVA Card. Pour ce faire, l'invention vise à rendre le code des applications indépendant de la protection. La spécification de la politique de protection, c'est-à-dire de la gestion des capacités, est séparée du code des applications.

- Permettre la coopération entre des applications dans la carte, mais également entre des applications dans la carte et des applications hors de la carte. Ceci nécessite de considérer le système d'exploitation dans la carte comme un milieu sécurisé et l'extérieur de la carte comme un milieu hostile.

L'atteinte de ces deux objectifs par l'invention apporte une grande simplicité de programmation du contrôle d'accès, ce qui réduit également le coût de développement et de maintenance du code, ainsi que les risques d'erreur de programmation de la protection.

Le premier objectif conduit à séparer le développement de l'application et la gestion des droits d'accès, simplifiant ainsi la complexité. Le programmeur d'applications programme des applications sans se soucier de la gestion des droits d'accès.

Celle-ci est spécifiée séparément dans un formalisme simple et très intuitif.

Pour le deuxième objectif, l'invention offre un modèle de gestion des droits d'accès uniforme alors que les contraintes sous-jacentes sont très différentes lorsque l'on se trouve dans la carte ou hors de la carte.

Les deux objectifs répondent à la même motivation consistant à masquer les complexités inhérentes à la protection et à la carte, et à simplifier la programmation d'applications coopérantes protégées.

Pour atteindre ces objectifs, l'invention fournit un procédé de génération d'applications caractérisé en ce qu'il comprend :

- une étape de développer une application comprenant un objet ou plusieurs objets, sans restriction d'accès ;
- une étape de définir des règles de droits d'accès à l'objet ou aux objets, compris au sein de l'application, depuis une seconde application ou depuis plusieurs autres applications ;
- une étape de transformer l'application comprenant le ou les objets par ajout à ladite application, de moyens de filtrage des accès audit objet ou auxdits objets pour mettre en œuvre un procédé de contrôle d'accès assurant la coopération des applications;
- une étape d'implanter l'application transformée au sein d'un moyen de traitement de données.

Selon une première réalisation, le moyen de traitement de données est inclus dans une carte à puce.

Selon une seconde réalisation, le moyen de traitement de données est inclus dans une station d'accueil de la carte à puce.

En outre, selon l'invention, le procédé de contrôle d'accès entre deux applications coopérant chacune au moyen de capacités sur des objets appartenant à l'autre application, les applications coopérant à travers au moins un système d'exploitation, est caractérisé par l'étape suivante :

lorsqu'à l'une des applications, dite application demandeur d'accès, est donné un accès à un objet appartenant à l'autre application, dite application fournisseur d'accès ;

créer deux capacités respectivement dans lesdites applications demandeur et fournisseur d'accès, en tant qu'objets ;

la capacité créée dans l'application fournisseur d'accès pour limiter l'accès audit objet et ;

la capacité créée dans l'application demandeur d'accès pour associer l'application demandeur d'accès à la capacité créée dans l'application fournisseur d'accès.

Selon un autre aspect de l'invention, lors de l'accès à un objet appartenant à l'une des applications, si un deuxième objet appartenant à l'une des applications est passé à cette application, le procédé comprend l'étape d'ajouter deux autres capacités respectivement dans les applications pour protéger l'accès au deuxième objet.

En pratique, la capacité d'accès au deuxième objet appartenant à l'une des applications est passée en paramètre ou en résultat à l'autre application.

Selon une première réalisation, les applications peuvent être implantées dans un moyen traitement de données commun, par exemple dans une carte à puce ou un terminal. Dans le cas d'une carte à puce, les

vérifications du code chargé dans la carte à puce permettent d'assurer qu'une capacité ne peut être créée par un programmeur pirate.

Selon une deuxième réalisation, les applications sont implantées dans deux moyens de traitement de données distants échangeant des messages d'accès à des objets distants. L'étape d'ajouter deux autres capacités peut comprendre alors de préférence le stockage d'un mot secret dans les deux autres capacités, passé à celles-ci par les deux capacités précédemment créées à l'étape de créer, afin de valider l'accès au deuxième objet.

Les moyens de traitement de données peuvent être inclus respectivement dans une carte à puce et une station d'accueil de la carte à puce, ou dans deux cartes à puce distinctes, ou dans deux contrôleurs d'une carte à puce ou d'un terminal.

D'autres caractéristiques et avantages de la présente invention apparaîtront plus clairement à la lecture de la description suivante de plusieurs réalisations préférées de l'invention en référence aux dessins annexés correspondants dans lesquels :

- la figure 1 montre une matrice de droits d'accès déjà commentée ;
- la figure 2 est un bloc-diagramme montrant des interfaces entre une application banque et une application client ;
- la figure 3 est un bloc-diagramme montrant des interfaces de deux applications coopérantes selon l'invention ;
- la figure 4 est un bloc-diagramme montrant l'implantation d'objets filtres entre deux applications coopérantes à une étape initiale de procédé selon l'invention ;

- la figure 5 est un bloc-diagramme montrant l'adjonction de deux filtres lors de l'appel par une application de la méthode sur un premier objet dans une autre application, selon une première réalisation du procédé de l'invention ; et

- la figure 6 est un bloc-diagramme analogue à la figure 5, montrant l'adjonction de deux filtres lors de l'appel par une application dans une station d'appel de la méthode sur un premier objet dans une autre application dans une carte à puce, selon une deuxième réalisation de l'invention.

Le concept général sous-jacent à l'invention est la gestion de capacités, c'est-à-dire la gestion de droits d'accès élémentaires, de façon séparée d'une application.

Lorsque deux applications coopèrent, cette coopération suit un protocole de coopération pré-établi. Ce protocole de coopération prend généralement la forme d'une interface commune permettant aux applications de s'appeler. Plus précisément, dans le contexte de la JAVA Card, chaque application qui désire appeler une autre application le fait en utilisant une interface JAVA qui est celle qu'est sensée fournir l'application appelée.

En référence à l'exemple illustré à la figure 2, une banque BA, c'est-à-dire une application banque dans un serveur, gère des comptes pour des clients. Lorsqu'un client CL se connecte à la banque à travers un objet Guichet OG, la banque lui retourne une référence Ref sur son objet Compte en banque OC pour qu'il puisse lire l'état de son compte. Chacune des applications banque et client connaît les interfaces de coopération IG et IC qui sont celles des objets Guichet et Compte. L'interface de l'objet Guichet IG

permet au client de se connecter à la banque en donnant son nom et un code d'accès, tel qu'un code d'identité personnel PIN (Personal Identity Number), et de lui retourner la référence Ref à l'objet Compte OC. L'interface de l'objet Compte IC fournit deux méthodes respectivement pour lire et écrire le solde du compte en banque, la syntaxe utilisée étant celle du langage JAVA.

Les besoins en termes de protection dans cet exemple sont présentés ci-après. La banque possède tous les droits sur ses propres objets. Mais il est impensable que la banque accorde tous les droits à ses clients. Un client peut lire le solde de son compte en banque, mais ne doit pas être autorisé à écrire arbitrairement le solde de son compte. C'est pourquoi, dans un schéma de protection par capacités, la banque retourne en résultat de l'opération de connexion une capacité correspondant à la référence sur l'objet Compte, mais avec des droits qui ne permettent que l'appel de l'opération de lecture de compte. La banque qui possède tous les droits sur ses propres objets, y compris l'objet Compte, crée une capacité n'autorisant que la lecture sur cet objet et retourne cette capacité à son client.

Etant donné que le premier objectif visé par l'invention est de séparer la définition de la politique de protection et le code de l'application, l'invention vise plus particulièrement à fournir des règles d'échange de capacité entre les applications au niveau des interfaces utilisées pour coopérer.

Dans l'exemple montré à la figure 2, la fourniture du contrôle d'accès par programmation est située au niveau de l'interface de l'objet Guichet OG de la banque BA. Selon un premier aspect de l'invention, cet outil de programmation spécifie dans

les interfaces utilisées la capacité qu'il faut transférer lors d'une transmission de paramètre entre des applications. On obtient une interface "protégée" appelée vue (*view* en anglais) qui prend la forme suivante en langage JAVA, le mot "*String*" désignant un objet de chaîne de caractères :

```
view guichet {
    Compte_client connexion (String nom, String pin-code) ;
}
view Compte_client {
    Solde lire () ;
    NOT void écrire (Solde s) ;
}
```

La spécification de ces deux vues indique que la banque BA ne laisse sortir que des capacités permettant de lire le solde des comptes. Le code de l'application banque et de l'application client reste ainsi totalement indépendant de la protection.

De façon plus générale, l'outil de programmation permet à chacune des applications, l'application banque et l'application client selon l'exemple précédent, de définir ses propres règles de protection. Lorsqu'une application AA possède une capacité vers un objet OB d'une application AB comme montré à la figure 3, la capacité inclut les règles de protection définies par l'application AA et regroupées dans une interface protégée "vue" IA et les règles de protection définies par l'application AB et regroupées dans une interface protégée "vue" IB. La vue IB a deux rôles : limiter les méthodes que l'application AA peut appeler sur l'objet OB et associer la vue choisie par l'application AB aux capacités entrantes dans et sortantes de

l'application AB lorsque l'objet OB est appelé. La vue IA ne remplit que le deuxième rôle pour les capacités entrantes dans et sortantes de l'application AA.

Ainsi, pour chaque capacité échangée en paramètre de l'appel depuis l'application AA vers l'application AB, ou depuis l'application AB vers l'application AA, la vue IA associe une vue IA' à cette capacité et la vue IB associe une vue IB' à cette capacité.

Un autre aspect de l'invention est l'intégration de l'outil de programmation dans le contexte de la carte à puce. Dans le contexte d'une carte à puce ouverte, des applications sont chargées dans la carte. Ces applications, dites applications internes, interagissent entre elles, mais également avec des applications, dites applications externes, exécutées dans une station d'accueil, telle qu'un terminal bancaire, un point de vente ou un terminal radiotéléphonique mobile, dans laquelle la carte est insérée. Ceci implique que des capacités sont échangées entre des applications internes et des application externes. L'outil de programmation associé à l'invention est mis en place en considérant les parties suivantes de ce contexte liées à la carte et à la station d'accueil :

- La carte à puce JAVA Card est un milieu protégé dans le sens où le code JAVA chargé dans la carte à puce est vérifié avant son chargement effectif. Cette vérification vise à assurer que le code à charger possède bien certaines propriétés du code JAVA, principalement liées à la sécurité. Le langage JAVA ne permet pas de manipuler directement des adresses et donc d'écraser arbitrairement de la

mémoire, ce qui assure un certain degré de sécurité lorsque différents programmes sont hébergés dans la même machine virtuelle JAVA. L'implantation du schéma de protection de l'invention tient compte de ce contexte interne à la carte pour l'implantation des capacités dans la carte.

- La station d'accueil dans laquelle la carte est insérée n'est pas forcément un milieu sécurisé. En effet, la carte peut être insérée dans n'importe quelle station d'accueil, et la station d'accueil peut très bien envoyer des données fabriquées pour tromper la carte. Lorsqu'elles sont propagées hors de la carte, les capacités sont selon l'invention protégées par des secrets qui permettent de vérifier la validité des capacités utilisées par des applications externes à la carte.

L'invention porte ainsi sur un outil de programmation pour programmer la gestion des droits d'accès entre des applications coopérantes s'exécutant dans la carte à puce ou dans la station d'accueil. La définition des règles de gestion des droits d'accès est séparée du code des applications, ce qui confère une plus grande clarté. L'intégration dans le contexte de la carte à puce JAVA Card nécessite de gérer différemment les capacités dans la carte à puce et à l'extérieur de celle-ci.

L'outil de programmation associé à l'invention spécifie les règles de protection d'une application séparément du code de l'application.

Il est supposé qu'une première application Aa écrite en langage JAVA coopère avec une deuxième application Ab également écrite en langage JAVA à travers une interface de coopération Iab implantée dans un unique moyen de traitement de données, par

exemple une carte à puce avec un système d'exploitation spécifique et la machine virtuelle JAVA :

```
interface Iab {
     void meth1 (I1 p1) ;
     I2 meth2 ();
     void meth3 ();
}
```

L'interface Iab contient les définitions de trois méthodes meth1, meth2 et meth3. La méthode meth1 prend en paramètre p1 une référence à un objet de type interface I1 et ne retourne aucun résultat. La méthode meth2 ne prend aucun paramètre et retourne un résultat de type interface I2. La méthode meth3 ne prend aucun paramètre et ne retourne aucun résultat.

Chaque application spécifie alors des interfaces de protection appelées vues, par exemple les vues suivantes Iab_V, I1_V1 et I2_V2 :

```
view Iab_V{
      void meth1 (I1_V1 p1);
      I2_V2 meth2 ();
      NOT void meth3 ();
}
```

Lorsque l'application Aa possède une capacité sur un objet appartenant à l'application Ab, les applications Aa et Ab ont la possibilité de spécifier la protection à associer à cette capacité en associant une vue à cette capacité.

La vue Iab_V indique que seules les méthodes meth1 et meth2 sont autorisées. Elle indique aussi que si la méthode meth1 est appelée, alors

l'application, qui peut être une application appelante ou une application appelée, se protège en associant à la capacité passée en paramètre la vue I1_V1. Enfin, elle indique que si la méthode meth2 est appelée, alors l'application se protège en associant à la capacité passée en résultat la vue I2_V2.

Lorsque l'application Aa possède une capacité vers un objet de l'application Ab, la vue que l'application Aa associe à cette capacité permet à l'application Aa de contrôler les capacités qui entrent dans et sortent de celle-ci, c'est-à-dire d'associer à ces capacités une vue. Réciproquement, la vue que l'application Ab associe à cette capacité permet à l'application Ab de contrôler les capacités qui entrent dans et sortent de celle-ci, mais aussi de limiter les méthodes qui peuvent être appelées sur l'objet de l'application Ab.

En général, la première exportation d'une capacité d'accès depuis l'application Ab vers l'autre application Aa et la première importation de cette capacité par l'autre application Aa, passent par un serveur de nom. L'application Ab exporte la capacité d'accès en l'associant à un nom symbolique, tel qu'une chaîne de caractères, et l'application Aa importe la capacité d'accès exportée en interrogeant le serveur de nom avec le nom symbolique. L'application Ab exporte la capacité en spécifiant explicitement la vue que l'application Ab y associe. L'application Aa importe la capacité exportée en spécifiant explicitement la vue que l'application Aa y associe. Pour l'application Aa comme pour l'application Ab, la vue spécifiée indique pour tous les échanges de capacité en paramètre découlant de ce

premier échange, les vues qui seront associées à ces capacités passées en paramètre.

Il en résulte que, excepté ce premier échange de message de capacité d'accès, la politique de protection de chaque application selon l'invention, c'est-à-dire la manière d'échanger les capacités, est spécifiée au niveau des interfaces et n'est pas noyée dans le code de l'application.

L'implantation de la politique de protection selon l'invention repose sur la notion d'objets filtres, "illustrant" des objets capacités (signifiant aptitudes ou facultés, ou en anglais "capabilities"), qui sont insérés entre les applications Aa et Ab. Pour chaque vue définie par une application, une classe filtre est générée et une instance de cette classe est insérée à l'exécution dans la chaîne d'accès à un objet dont une capacité est exportée. Comme montré à la figure 4, si à une étape initiale E0 un accès à un objet Ob appartenant à l'application Ab est donné à l'application Aa, la vue de l'application Aa pour la capacité de cet accès est implantée par un filtre Fa et la vue de l'application Ab pour cette capacité est implantée par un filtre Fb.

Une classe filtre définit toutes les méthodes déclarées dans la vue que le filtre implante. Son rôle est de retransmettre l'appel de méthode vers son successeur dans la chaîne d'accès à l'objet. Selon l'exemple montré à la figure 4, le filtre Fa retransmet l'appel vers le filtre Fb et le filtre Fb vers l'objet Ob.

Par contre, une classe filtre implante la politique de protection définie par la vue à partir de laquelle elle est générée : le filtre Fb ne laisse

passer que les méthodes autorisées par la vue de l'application Ab ; et les filtres Fa et Fb implantent aussi l'association des vues aux capacités passées en paramètre. Selon l'exemple de vues ci-dessus, la vue Iab_V indique l'association de la vue I1_V1 au paramètre p1 de la méthode meth1. L'association de la vue à la capacité est implantée en insérant dans la chaîne d'accès à l'objet passé en paramètre un objet filtre correspondant à la vue.

En référence à la figure 5, pour deux applications Aa et Ab implantées dans une carte à puce CP, l'application Aa possède à l'étape initiale E0 une capacité vers l'objet Ob de l'application Ab, et comme dans la figure 4, les accès ultérieurs à l'objet Ob sont protégés par le filtre Fa(Ob) de l'application Aa et par le filtre Fb(Ob) de l'application Ab. A une première étape E1, lors de l'appel par l'application Aa de la méthode meth1 sur l'objet Ob appartenant à l'application Ab, c'est-à-dire lors de l'accès à l'objet Ob, une capacité sur un objet Oa appartenant à l'application Aa est passée en paramètre à l'autre application Ab. Pour mettre en oeuvre la protection spécifiée par l'application Aa dans sa vue à une deuxième étape E2, le filtre Fa(Ob) ajoute le filtre Fa(Oa) et le passe en paramètre de la méthode meth1 à la place de la référence directe à l'objet Oa. De même, pour mettre en oeuvre la protection spécifiée par l'application Ab dans sa vue, le filtre Fb(Ob) ajoute le filtre Fb(Oa) et le passe en paramètre de la méthode meth1 à la place du paramètre reçu.

Les objets filtres Fa(Ob) et Fb(Ob), lorsqu'ils sont appelés, sont donc chargés d'installer des objets filtres Fa(Oa) et Fb(Oa) pour les références passées en paramètre ; en d'autres termes, deux

capacités illustrées par les filtres Fa(Oa) et Fb(Oa) protégeant l'accès à l'objet Oa sont ajoutées respectivement dans les applications Aa et Ab.

Il est indiqué ci-après un exemple de classe filtre générée F_Iab_V pour la vue Iab_V donnée auparavant, rappelée ci-après :

```
view Iab_V {
        void meth1 (I1_V1 p1);
        I2_V2 meth2 ();
        NOT void meth3 ();
}
```

Pour les vues I1_V1 et I2_V2 sont créées respectivement des classes filtres F_I1_V1 et F_I2_V2. Il est supposé que les deux applications Aa et Ab qui coopèrent se trouvent dans le même environnement JAVA et les lignes suivantes sont encore écrites en code JAVA, dans lesquelles le mot-clé *public* signifie que la méthode déclarée suivante est accessible à toutes les classes, le mot-clé *void* signifie que la méthode déclarée suivante une fois exécutée ne retourne aucun résultat, et le mot-clé *new* désigne un opérateur de création de classe :

```
public class F_Iab_V implements Iab {
 Iab obj;

 public F_Iab_V (Iab o) {
      obj = o;
 }

 public void meth1 (I1_V1 p1) {
      obj.meth1 (new F_I1_V1 (p1));
```

```
    }

    public I2_V2 meth2 () {
        return (new F_I2_V2 (obj.meth2()));
    }

    public void meth3 () {
        // propager une exception
    }
}
```

La variable obj est une référence à l'entité suivante dans le chemin d'accès à l'objet, le deuxième filtre ou l'objet réel. Elle est utilisée pour retransmettre l'appel s'il est autorisé.

La méthode F_Iab_V est la méthode constructeur de la classe filtre. Elle initialise la variable obj. Lorsque l'application Aa importe une capacité du serveur de noms et veut lui associer la vue Iab_V, elle appelle la méthode constructeur en lui passant en paramètre la référence JAVA reçue.

La méthode meth1 retransmet un appel qui est donc autorisé, mais elle doit associer à la capacité passée en paramètre p1 la vue I1_V1. La méthode meth1 crée par l'opérateur *new* un filtre F_I1_V1 à partir du paramètre reçu, puis retransmet l'appel en passant en paramètre p1 la référence au filtre créé.

La méthode meth2 retransmet l'appel et elle reçoit en retour une capacité. Elle doit y associer la vue I2_V2. La méthode meth2 crée donc une instance de la classe F_I2_V2 à partir du paramètre reçu et retourne par l'instruction *return* la référence à l'objet filtre créé en retour de la méthode meth2.

La méthode meth3 ne retransmet pas l'appel puisqu'il n'est pas autorisé, et propage donc une exception.

L'implantation lorsque les applications coopérantes Aa et Ab sont toutes les deux dans la carte à puce respecte les principes décrits ci-dessus. Par contre, lorsqu'une des deux applications se trouve hors de la carte, l'implantation est sensiblement différente.

Il est rappelé que les appels de méthodes entre une station d'accueil SA, telle qu'un terminal, et une carte à puce CP comme montré schématiquement à la figure 6 sont implantés à partir de messages, appelés unités de données de protocole applicatif APDU, entre la station d'accueil SA et la carte à puce CP, ces appels de méthode n'étant effectués que suivant le sens de la station d'accueil vers la carte. En effet, la station d'accueil et la carte à puce, ou en variante deux cartes à puce ou deux contrôleurs dans une carte à puce ou un terminal, comprennent des microcontrôleurs constituant des moyens de traitement de données qui sont respectivement maître et esclave et qui dialoguent selon un protocole d'échange de données asynchrone qui oblige la station d'accueil à interroger périodiquement la carte pour que celle-ci déclenche en réponse une action dans la station d'accueil.

En variante, la station d'accueil dans la suite est remplacée par une autre carte à puce, c'est-à-dire les applications coopérantes Aa et Ab sont implantées respectivement dans deux cartes à puce, ou plus généralement dans deux contrôleurs.

Le protocole d'échange de données asynchrones implique que, pour un appel relatif à un objet Ob1

depuis l'application Aa dans la station d'accueil SA vers l'application Ab dans la carte CP à laquelle appartient l'objet Ob1, lorsqu'il y a retransmission d'un appel entre deux objets filtres Fa(Ob1) et Fb(Ob1), relatifs à l'objet Ob1, cette retransmission de l'appel prenne la forme d'un échange de message entre la station d'accueil et la carte. Une station d'accueil d'origine inconnue (pirate) est alors capable d'établir un message correspondant à un appel de méthode bien qu'elle ne soit pas réellement autorisée à réaliser cet appel de méthode. Ceci revient à créer une capacité ce qui n'est pas possible dans le schéma de protection selon l'invention.

Pour protéger les capacités contre ces attaques, des secrets, tels que mots de passe mdp, éventuellement basés sur des méthodes de chiffrement, sont utilisés.

Comme montré à la figure 6, lorsqu'à une première étape E3 l'application Aa dans la station d'accueil SA appelle la méthode meth2 sur un objet Ob1 appartenant à l'autre application Ab dans la carte CP, c'est-à-dire lors de l'accès à l'objet Ob1, et lorsqu'à une deuxième étape E4 le résultat de cet appel de méthode retourne une capacité d'accès sur un objet Ob2 appartenant à l'application Ab, le filtre Fb(Ob1) crée le filtre Fb(Ob2) dans le système d'exploitation de la carte CP pour cette capacité d'accès. Selon l'invention, le filtre Fb(Ob1) génère alors un secret, tel qu'un mot de passe mdp qui est stocké dans le filtre Fb(Ob2) et retourné au filtre Fa(Ob1) qui le stocke également. Ainsi, lorsque le filtre Fa(Ob1) crée le filtre Fa(Ob2) dans la station d'accueil SA pour la capacité d'accès retournée, le filtre Fa(Ob1) passe au filtre Fa(Ob2) le mot de

passe mdp qui y est stocké. En d'autres termes, l'accès à l'objet Ob2 est protégé dans les applications Aa et Ab respectivement par deux capacités ajoutées illustrées par les filtres Fa(Ob2) et Fb(Ob2).

Lorsqu'à l'étape E4 cette capacité retournée en paramètre est utilisée pour appeler une méthode sur l'objet Ob2, l'appel entre les filtres Fa(Ob2) et Fb(Ob2) inclut le mot de passe dans le message APDU, ce qui permet au filtre Fb(Ob2) de vérifier que la capacité d'accès à l'objet Ob2 est valide.

L'invention crée ainsi par nommage une correspondance entre un objet et une capacité illustrée par un filtre, gérés par les systèmes d'exploitation dans les moyens de traitement de données, tels que station d'accueil et carte à puce. Si un objet est supprimé dans une application, le système d'exploitation respectif détruit le filtre correspondant.

Ainsi, la coopération entre une application Ab dans la carte avec une application quelconque nécessite de gérer des capacités pouvant prendre deux formats, avec mot de passe si l'application quelconque est exportée hors de la carte et sans mot de passe si elle reste dans la carte.

Le schéma de protection ci-dessus utilise des références, sortes de pointeurs, à des objets JAVA qui sont presque des capacités. En effet, étant donné que le langage JAVA est sûr, il n'est pas possible dans un programme JAVA d'établir une référence à un objet et d'appeler une méthode sur cet objet. Ceci implique que si un objet O1 crée un objet O2, l'objet O2 n'est pas accessible à partir des autres objets de l'environnement JAVA, tant que l'objet O1 ne transmet

pas explicitement à l'objet O2 une référence à ces autres objets. Cette transmission de référence ne peut se faire que par passage de paramètre lorsqu'un objet appelle l'objet O1 ou lorsque l'objet O1 appelle un autre objet.

REVENDICATIONS

1 - Procédé de contrôle d'accès entre deux applications (Aa, Ab) coopérant chacune au moyen de capacités sur des objets appartenant à l'autre application, les applications coopérant à travers au moins un système d'exploitation et étant implanté dans un moyen de traitement de données (CP), caractérisé par l'étape suivante :

lorsqu'à l'une (Aa) des applications, dite application demandeur d'accès, est donné un accès à un objet (Ob ; Ob1) appartenant à l'autre application (Ab), dite application fournisseur d'accès ;

créer (E0) deux capacités (Fa(Ob), Fb(Ob) ; Fa(Ob1), Fb(Ob1)) respectivement dans lesdites applications demandeur et fournisseur d'accès, en tant qu'objets ;

la capacité créée dans l'application fournisseur d'accès (Ab) pour limiter l'accès audit objet (Ob) et ;

la capacité créée dans l'application demandeur d'accès (Aa) pour associer l'application demandeur d'accès à la capacité créée dans l'application fournisseur d'accès (Ab).

2 - Procédé conforme à la revendication 1, comprenant lors de l'accès (E1; E3) à un objet (Ob ; Ob1) appartenant à l'une (Ab) des applications, si un deuxième objet (Oa ; Ob2) appartenant à l'une (Aa ; Ab) des applications est passé à cette application, l'étape d'ajouter (E2, E4) deux autres capacités (Fa(Oa), Fb(Oa) ; Fa(Ob2), Fb(Ob2)) respectivement dans les applications (Aa, Ab) pour protéger l'accès au deuxième objet.

3 - Procédé conforme à la revendication 2, selon lequel la capacité d'accès au deuxième objet (Oa ;

Ob2) appartenant à l'une des applications est passée en paramètre ou en résultat à l'autre application.

4 - Procédé conforme à l'une quelconque des revendications 1 à 3, caractérisé par l'exportation d'une capacité depuis une application (Ab) vers l'autre application (Aa) en l'associant à un nom symbolique, et l'importation de la capacité d'accès exportée dans l'autre application en interrogeant un serveur de nom avec le nom symbolique.

5 - Procédé conforme à l'une quelconque des revendications 1 à 4, selon lequel les applications (Aa, Ab) sont implantées dans un moyen de traitement de données commun (CP).

6 - Procédé conforme à l'une quelconque des revendications 1 à 4, selon lequel les applications (Aa, Ab) sont implantées respectivement dans deux moyens de traitement de données distants (SA, CP) échangeant des messages d'accès à des objets distants.

7 - Procédé conforme à la revendication 6 lorsqu'elle dépend de la revendication 2 ou 3, selon lequel l'étape d'ajouter deux autres capacités (E4) comprend le stockage d'un mot secret (mdp) dans les deux autres capacités (Fa(Ob2), Fb(Ob2)) passé à celles-ci par les deux capacités (Fa(Ob1), Fb(Ob1)) précédemment créées.

8 - Procédé conforme à la revendication 6 ou 7, selon lequel les moyens de traitement de données sont inclus respectivement dans une carte à puce (CP) et une station d'accueil (SA) de la carte à puce.

9 - Procédé conforme à la revendication 6 ou 7, selon lequel les moyens de traitement de données sont inclus respectivement dans deux cartes à puce.

10 - Procédé de génération d'applications caractérisé en ce qu'il comprend :

- une étape de développer une application (Ab) comprenant un objet (Ob) ou plusieurs objets (Ob, Ob1), sans restriction d'accès ;
- une étape de définir des règles de droits d'accès à l'objet (Ob) ou aux objets (Ob, Ob1) compris au sein de l'application depuis une seconde application (Aa) ou depuis plusieurs autres applications ;
- une étape de transformer l'application (Ab) comprenant le ou les objets (Ob, Ob1) par ajout à ladite application (Ab), de moyens de filtrage (Fa, Fb) des accès audit objet (Ob) ou auxdits objets (Ob, Ob1) pour mettre en œuvre le procédé de contrôle d'accès selon les revendications 1 à 4;
- une étape d'implanter l'application transformée au sein d'un moyen de traitement de données (CP, SA).

11 - Procédé conforme à la revendication 10 selon lequel le moyen de traitement de données est inclus dans une carte à puce.

12 - Procédé conforme à la revendication 10 selon lequel le moyen de traitement de données est inclus dans une station d'accueil (SA) de la carte à puce.

FIG.1

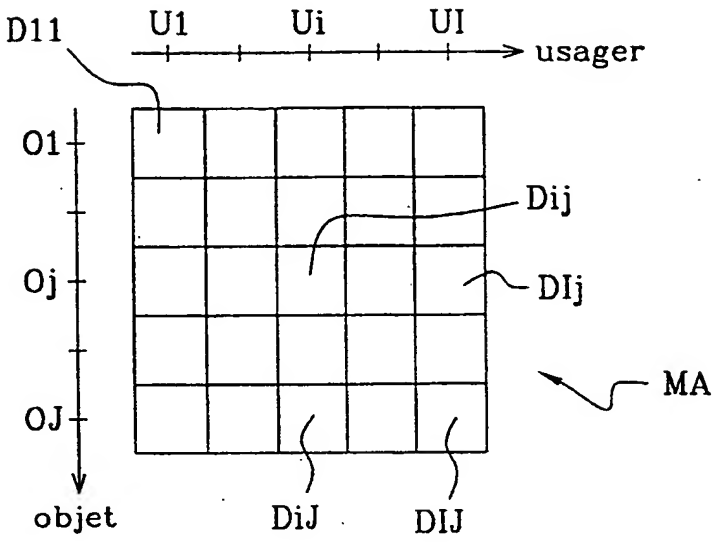

D11
U1
Ui
UI
usager
O1
Oj
OJ
objet
Dij
DIj
MA
DiJ
DIJ

FIG.2

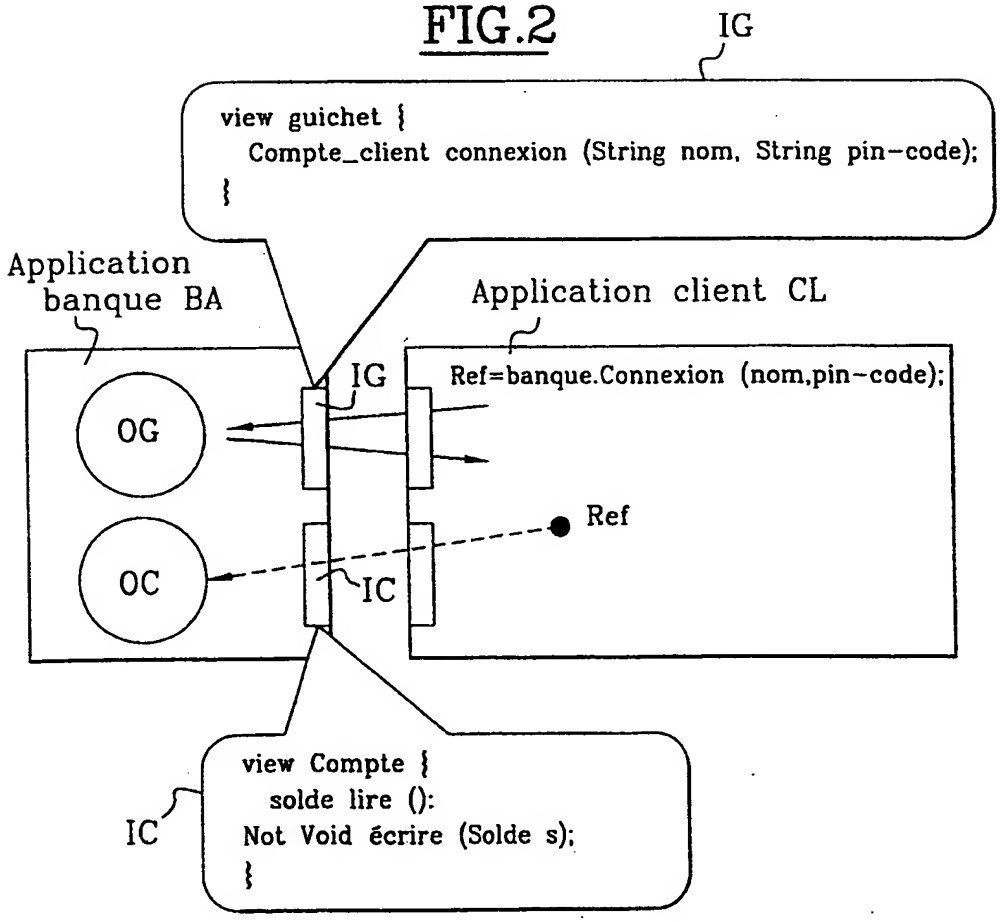

IG
view guichet {
Compte_client connexion (String nom, String pin-code);
}
Application banque BA
Application client CL
IG
Ref=banque.Connexion (nom,pin-code);
OG
OC
Ref
IC
IC
view Compte {
solde lire ():
Not Void écrire (Solde s);
}

FIG.3

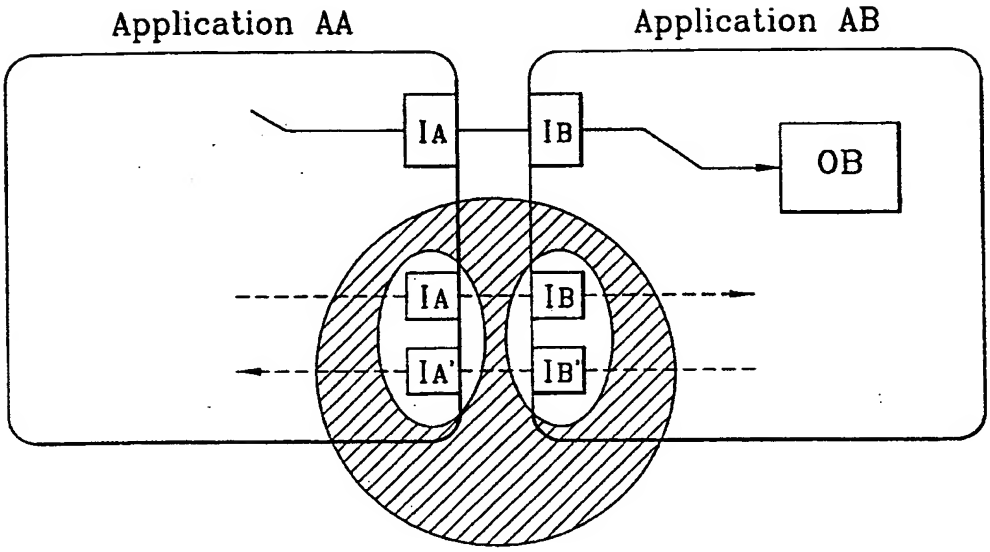

Application AA
Application AB
IA
IB
OB
IA
IB
IA'
IB'

FIG.4

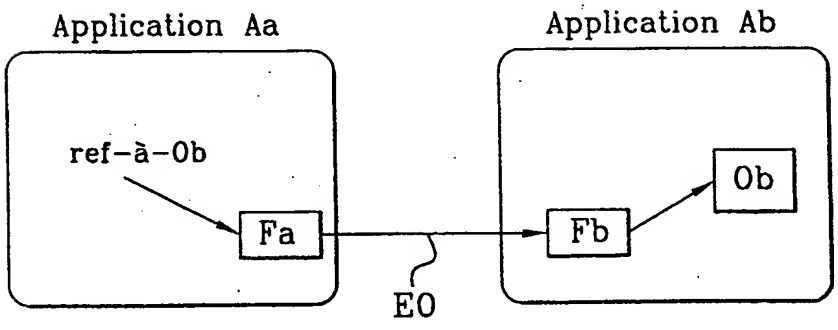

Application Aa
Application Ab
ref-à-Ob
Fa
Fb
Ob
EO

FIG.5

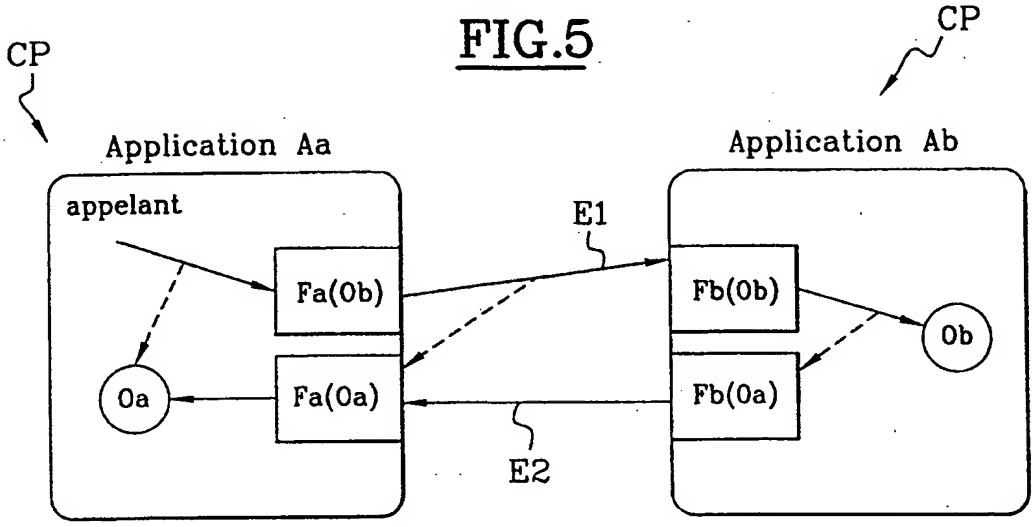
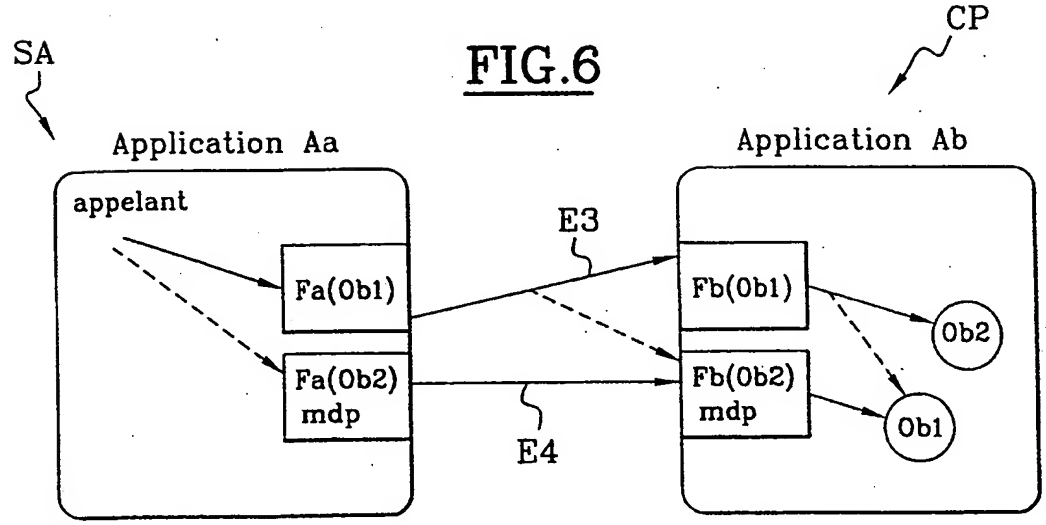

CP
CP
Application Aa
Application Ab
appelant
E1
Fa(0b)
Fb(0b)
0b
0a
Fa(0a)
Fb(0a)
E2

FIG.6


SA
CP
Application Aa
Application Ab
appelant
E3
Fa(0b1)
Fb(0b1)
0b2
Fa(0b2)
mdp
Fb(0b2)
mdp
0b1
E4